MINI PROJECT REPORT

MECHATRONICS

SYSTEM

INTEGRATION

MCTA 3203

SEMESTER 2 2024/2025

GROUP: 6

	NAME	MATRIC NUMBER
1	MUHAMMAD NABIL IMAN BIN ABD RAHMAN	2313551
2	MUHAMMAD NAZHAN BIN MOHAMED NADZRI	2313703
3	MUHAMMAD HAZIQ AJMAL BIN MD KAMAL	2312821

Abstract

This project presents the development of a Smart Laundry System that integrates Arduino-based hardware and Firebase-based software to automate and streamline the laundry process in shared environments such as dormitories or hostels. The system utilizes an **Arduino Uno** microcontroller, a **Pixy CMUcam5 image sensor** for color identification, and a **360 servo** to simulate a washing machine. It incorporates an **ESP32** for Wi-Fi connectivity, enabling communication with a **Firebase Realtime Database** where user data, session status, and credit balances are stored and updated in real time.

A set of **push buttons** allows users to interact with the system, while a **20x4 LCD display** provides clear feedback on session status and balance information. When a user is identified, the system checks their balance, deducts the required amount, and starts the motor to simulate a washing session. This project combines computer vision, cloud computing, and embedded systems to offer a secure, automated, and user-friendly solution to common problems in communal laundry usage.

Introduction

Laundry management in shared environments such as dormitories, hostels, and apartments often faces challenges such as machine availability, user accountability, and misuse of resources. The traditional method of managing laundry machines manually or via simple timers can be inefficient and inconvenient, leading to wasted time and frustration among users.

To address these challenges, this project introduces a **Smart Laundry System** that integrates both hardware and software components to automate and digitize the process. At the core of the system is an **Arduino Uno** microcontroller that interfaces with various hardware components including a **360 servo** (simulating a washing machine motor), **push buttons**, a **20x4 LCD display**, and a **Pixy CMUcam5 image sensor**. The system also incorporates **Firebase** as a real-time cloud database to manage user data such as balance and session activity, and an **ESP32** module to enable internet connectivity for the Arduino.

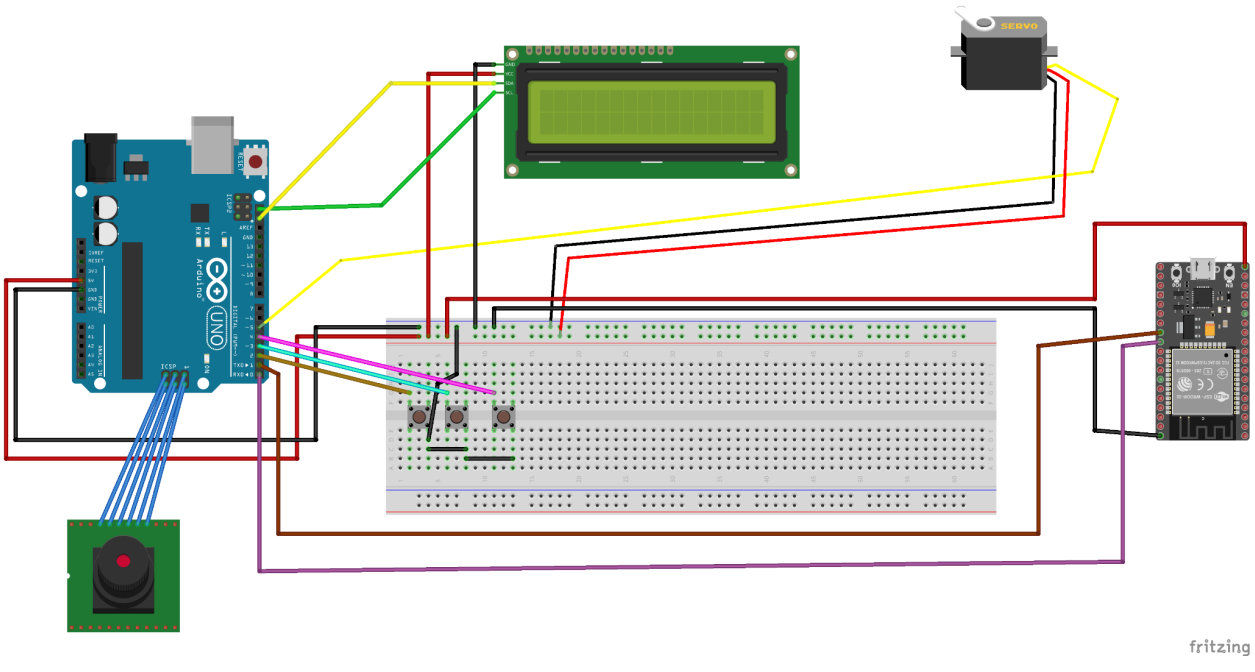
The system is designed to identify clothes using colored objects (via PixyCam), deduct credit balance before activating the machine, and display relevant information to the user. This not only encourages proper use of the machine but also provides real-time updates and control via cloud services. Ultimately, the Smart Laundry System aims to enhance efficiency, transparency, and user satisfaction in shared laundry spaces.

Materials and Equipments

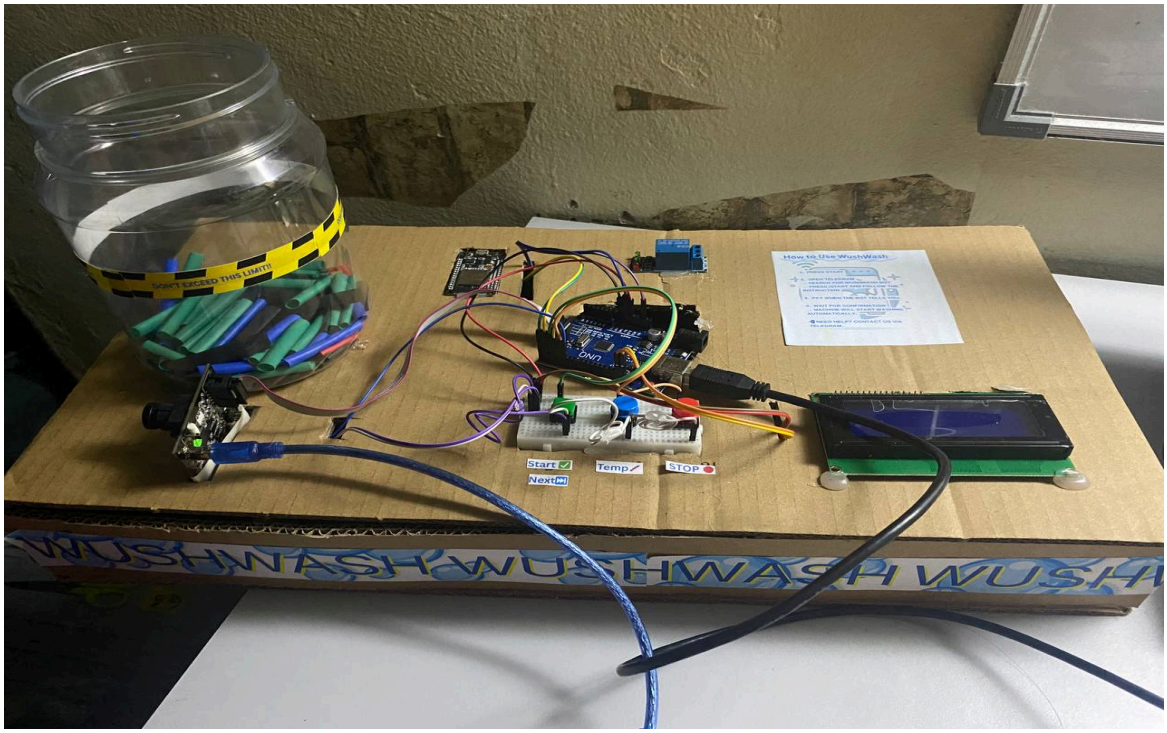
- 1x Arduino Uno
- 1x Micro 360 servo
- 1x 20x4 LCD Display
- 3x Push Buttons
- 1x Breadboard
- 1x Pixy CMUcam5 Image Sensor
- 1x Esp32
- 2x Usb Cable (A-B type)
- Jumper Wires
- 1x Usb Adapter
- Cardboards
- 1x Medium size jar

Experimental Setup

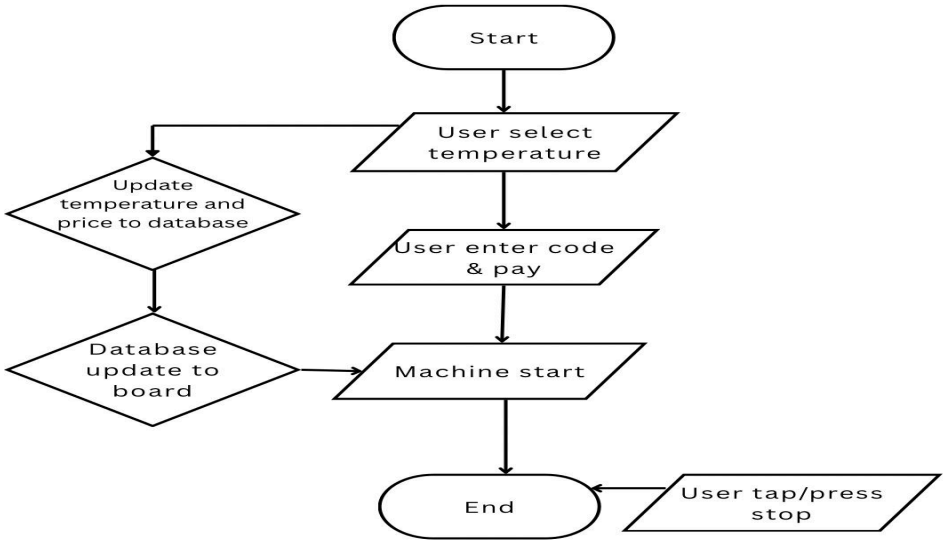
Wiring Diagram



Product Setup



Flow Chart



Methodology

1. Hardware Setup

The smart laundry system is developed using the following components:

- **Arduino Uno**: Acts as the central controller for user interaction, motor control, and communication with peripherals.
- **Micro 360 servo**: Simulates the washing machine motor. It runs based on session control.
- **20x4 LCD Display**: Displays user instructions, system status (e.g., "Idle", "Running", "Done"), and temperature information.

- **3x Push Buttons**: Each button represents a different wash cycle or operation (e.g., start, stop, mode selection).
- **Pixy CMUcam5 Image Sensor**: Detects clothes colors and specific it to whether green, blue or red.
- **ESP32**: Handles WiFi communication and synchronizes with Firebase for user authentication, balance updates, and session logging.
- **Breadboard & Jumper Wires**: Used to connect buttons, motor driver, and other components to the Arduino.
- **Power Supply**: Arduino is powered via USB from a PC or adapter. It also powers the ESP32 via 5V output. PixyCam is powered via its own USB cable.

2. Software Setup

- **Arduino Sketch**:
    - Contains logic to read button presses.
    - Sends session control data to ESP32.
    - Displays information on the LCD.
    - Controls the 360 servo to simulate a laundry session.
- **PixyCam**:
    - The PixyCam is trained to detect red, green, and blue objects. It sends the detected color to the ESP32, which logs it to Firestore.
    - Connected to Arduino through SPI/I2C/UART (depending on your wiring choice).
- **ESP32 Firmware**:
    - Connects to Firebase using credentials and a token generated by a service account.

- Communicates with Arduino over serial or shared GPIO pins.
- Reads session codes, fetches or updates user balance, and updates session status in Firebase.

- **Google Apps Script**:
  - Handles Firebase REST API calls (read/write user, update balance, create sessions).
  - Generates access tokens from the service account.
  - Used to integrate with other platforms like Telegram bots for remote control or alerts.

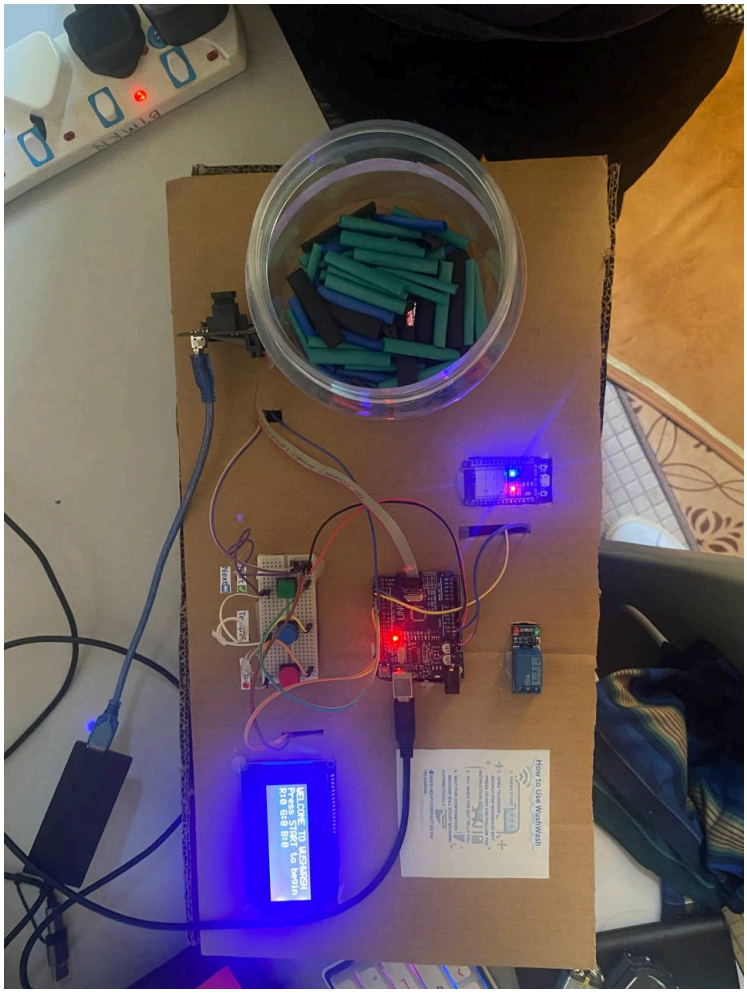
3. User Process

1. **Startup**:
   - Arduino powers on and initializes all components.
   - LCD displays a welcome message or prompts the user to scan a tag.
2. **User Identification and Color Identification:**
   - Users must first **register through the Telegram bot** by submitting their **matric number** and **creating a password**.
   - This data is stored in Firebase Firestore under a user document.
   - Once registered, the user can log in through the same bot interface by providing their matric number and password.
   - The bot checks credentials against Firebase.
   - **Only upon successful login** will the user be granted access to use the washing machine system.
   - After login, the user places a piece of clothing or colored tag in front of the **Pixy CMUcam5**.

- The PixyCam detects the color (limited to **red**, **green**, or **blue**) and sends it to the ESP32.
- The ESP32 records the detected color in the user's session document in **Firebase Firestore**.

3. **Session Selection**:
    - The user presses one of the push buttons to select a desired washing temperature condition.
    - The LCD displays the selected option and prompts confirmation.
4. **Session Start**:
    - If the user has enough balance, the session is started.
    - The 360 servo runs to simulate the machine operation.
    - ESP32 updates the session document in Firebase to "Running".
    - Balance is deducted automatically.
5. **Session End**:
    - After a predefined duration, the motor stops.
    - The session status is updated to "Completed" in Firebase.
    - The LCD displays a "Thank you" or "Session complete" message.
6. **Top-Up or Retry**:
    - If a user has insufficient balance, the telegram bot prompts the user to top-up.
    - Balance can be updated via the Firebase interface or a Telegram bot.

Results



Safety Issues and Precaution

The implementation of this smart laundry system, which involves both hardware and software integration, requires careful attention to safety to prevent damage to components, ensure user protection, and maintain system integrity during operation. This section outlines the main safety considerations and the corresponding precautions taken during the development and testing phases of the project.

First and foremost, **electrical safety** is a major concern due to the use of multiple powered components such as the Arduino Uno, ESP32 module, Pixy CMUcam5 image sensor, and the micro 360 servo. All components were connected using proper jumper

wires and a breadboard, with connections thoroughly checked to avoid short circuits. The Arduino board supplies power to the ESP32, and both are connected via USB cables. To prevent accidental electric shock or damage, all exposed conductive areas were kept away from the user's direct contact, and USB adapters with overcurrent protection were used to regulate voltage safely.

Another significant aspect is the **potential for power overload and overheating**, especially related to the 360 servo. Although the motor is used only to simulate the washing machine's motor, running it for extended periods can result in excessive heat buildup. As a precaution, the motor is only activated for brief demonstration periods, and it was ensured that the current draw remained within safe operational limits of both the Arduino and power supply. Monitoring the motor's temperature during use helped prevent any risk of overheating or component degradation.

The system also involved complex **wiring connections**, which introduced the risk of **component damage** through incorrect wiring or reverse polarity. To address this, all circuits were reviewed against the wiring diagram before powering the system. Components such as push buttons were wired with proper resistance where necessary, and tests were performed incrementally to confirm that each module functioned correctly before integration. Special care was taken when handling the PixyCam, as its USB communication requires stable power and a reliable signal connection, which if disrupted, could lead to sensor malfunction.

In terms of **user safety**, the system's design considered physical interaction points. The push buttons were positioned securely to avoid accidental disconnection, and the components were laid out on a non-conductive breadboard to prevent shorting. During development, the setup was placed on a stable, fire-resistant surface to avoid the risk of electrical fire. The PixyCam was mounted in a fixed position to avoid instability and ensure it did not obstruct the user's movement or line of sight.

Lastly, users were instructed on proper operation procedures to avoid mishandling. Clear labels for buttons and components were used where possible, and any interaction with the system was guided by a straightforward process to minimize misuse. As an added precaution, the system was never left powered when unattended, and unplugging was recommended after use.

In conclusion, the system's safety was prioritized through careful planning of electrical connections, load management, component positioning, and user interaction. These precautions ensured that the smart laundry system remained safe, reliable, and user-friendly throughout its development and testing stages.

Discussions

The development of the smart laundry system effectively demonstrates how microcontroller-based automation can enhance everyday tasks through simplicity, functionality, and connectivity. The integration of hardware components such as the Arduino Uno, ESP32, Pixy CMUcam5 image sensor, LCD display, push buttons, and a 360 servo provided a hands-on simulation of an intelligent laundry experience, while the software components facilitated real-time database communication and system responsiveness.

From a hardware perspective, the system's architecture was designed for modularity and ease of use. The Arduino Uno served as the central controller, handling button inputs, motor activation, and display outputs. The ESP32 module, powered directly by the Arduino, acted as the gateway for Firebase database communication, enabling wireless data exchange. The Pixy CMUcam5 played a unique role by detecting specific markers (such as visual labels), which simulated automated sessions for color detection for clothes. This innovative use of computer vision added an intelligent edge to the project.

The simulation of the washing machine motor using a micro 360 servo was crucial in demonstrating motor control functionality. Although not representative of a real appliance in terms of power, it allowed for safe and visible observation of start-stop behavior based on user interaction. The LCD display further enhanced usability by providing direct feedback to the user, such as session status and balance information.

Software-wise, the use of Firebase Firestore as the backend allowed for scalable and structured data management. The Google Apps Script code efficiently handled authentication via service accounts, managed user creation, balance top-ups, and session status updates. This integration proved that a lightweight but effective database solution can support IoT systems reliably. The system also implemented JWT-based access token generation, ensuring secure communication between the ESP32 and Firestore.

The user interaction process was designed to be intuitive. A user presses a button to initiate or top-up a session, and the system verifies the user's existence and balance before proceeding. If approved, the system activates the motor to simulate a washing session and updates the session status accordingly in the database. This linear process not only streamlines the experience but also demonstrates the principle of conditional logic based on cloud-stored data.

Challenges encountered during development included power management—particularly ensuring that the Arduino could reliably power the ESP32—and maintaining stable communication with Firebase due to token expiry. These were addressed through careful wiring, regulated power supply, and efficient token handling in the script.

Overall, the project successfully simulates a real-world laundry system with intelligent features using minimal hardware. It highlights how embedded systems, cloud computing, and computer vision can converge to deliver functional, low-cost smart systems for automation in daily life.

Conclusion

In conclusion, the smart laundry system prototype successfully demonstrates how embedded hardware and cloud-based services can be integrated to create a simple yet functional automated solution. Using components like the Arduino Uno, ESP32, Pixy CMUcam5, and a 360 servo, the system simulates real-world interactions such as user identification, session control, and balance management.

The combination of Firebase Firestore and Google Apps Script provides a reliable and scalable backend, enabling real-time communication and secure data handling. The user-friendly interface—enhanced by the LCD display and push buttons—offers a smooth experience that mirrors how a real smart appliance would function.

This project proves that even with basic components and limited resources, effective automation systems can be developed. It also lays the groundwork for further expansion, such as remote control via mobile apps, enhanced sensor integration, and energy efficiency optimization.

Recommendations

To further enhance the performance, reliability, and user experience of the smart laundry system, several improvements are recommended:

1. **Use of a More Powerful Motor Driver Circuit**
   The current setup uses a simple 360 servo for simulation. For actual application or better emulation, integrating a dedicated motor driver like the L298N or an H-bridge circuit will allow safer and more controlled operation of larger motors.

2. **Improve Power Management**
   Relying on USB power limits the system's flexibility. Incorporating a regulated external power supply (e.g., 9V-12V adapter with proper voltage regulators) will improve stability and allow independent operation without a PC.

3. **Secure Communication Protocols**
   While Firebase handles secure data transmission, adding authentication or

encryption between the ESP32 and Firebase can ensure user data and balance remain protected from tampering.

4. **User Interface Enhancements**
   Replacing the 16x2 LCD with a touchscreen or OLED display could provide a more interactive and modern interface for users, enabling dynamic visuals and easier input.

5. **Mobile App Integration**
   Building a companion mobile application using Flutter or MIT App Inventor can offer remote top-up, session monitoring, and status notifications to improve convenience.

References

Google Developers. (n.d.). *Firebase Documentation*. Retrieved from https://firebase.google.com/docs

Espressif Systems. (n.d.). *ESP32 Technical Reference Manual*. Retrieved from https://www.espressif.com/en/products/socs/esp32/resources

Charmed Labs. (n.d.). *Pixy CMUcam5 Vision Sensor*. Retrieved from http://charmedlabs.com/default/pixy-cmucam5/

OpenAI. (2025). *ChatGPT (June 11 version)* [Large language model]. https://chat.openai.com/

Appendix

Arduino Uno Code

```
#include <Pixy.h>
```

```
Pixy pixy;

/* cumulative totals that only go up */
int totalRed = 0, totalGreen = 0, totalBlue = 0, totalUnknown = 0;

/* previous-frame snapshot so we can detect "new arrivals" */
int prevRed = 0, prevGreen = 0, prevBlue = 0, prevUnknown = 0;

/* temporary per-frame counters */
int currRed = 0, currGreen = 0, currBlue = 0, currUnknown = 0;

unsigned long lastPixyUpdate = 0;
const unsigned long pixyInterval = 2000;   // poll every 2 s




#include <LiquidCrystal_I2C.h>
#include <SoftwareSerial.h>
#include <Servo.h>

LiquidCrystal_I2C lcd(0x27, 20, 4);
SoftwareSerial mySerial(0, 1); // RX, TX (Arduino -> ESP32)

const int startBtn = 2;
const int tempBtn = 3;
const int stopBtn = 4;
const int servoPin = 5;

Servo myServo;

String temps[] = {"COOL", "WARM", "HOT"};
int prices[] = {3, 4, 5};
int tempIndex = 0;

enum State {WELCOME, SELECT_TEMP, SHOW_CODE, WAITING, STARTED};
State currentState = WELCOME;

void setup() {
  pinMode(startBtn, INPUT_PULLUP);
  pinMode(tempBtn, INPUT_PULLUP);
  pinMode(stopBtn, INPUT_PULLUP);

  myServo.attach(servoPin);
  myServo.write(90); // STOP

  lcd.init();
  lcd.backlight();

  pixy.init();
```

```
  Serial.begin(9600);
  mySerial.begin(9600);

  showWelcome();
}

void loop() {
  // 🔴 STOP BUTTON
  if (digitalRead(stopBtn) == LOW) {
    Serial.println("Stop button pressed. Resetting.");
    resetAll();
    return;
  }

  // 📡 Listen to ESP32
  if (currentState != WELCOME && mySerial.available()) {
    String status = mySerial.readStringUntil('\n');
    status.trim();

    if (status == "paid") {
      if (currentState != STARTED) {
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Thank you!");
        lcd.setCursor(0, 1);
        lcd.print("Starting machine...");


        currentState = STARTED;
      }
    }

    else if (status == "unpaid" && currentState == STARTED) {
    resetAll();  // stop servo
}

  }

  // 🔁 State Machine
  switch (currentState) {
    case WELCOME:
      if (digitalRead(startBtn) == LOW) {
        delay(200);
        currentState = SELECT_TEMP;
        showSelection();
        updatePixyDisplay();
      }
      break;

    case SELECT_TEMP:
      if (digitalRead(tempBtn) == LOW) {
        tempIndex = (tempIndex + 1) % 3;
        delay(200);
        showSelection();
        updatePixyDisplay();
      }

      if (digitalRead(startBtn) == LOW) {
        delay(200);
        currentState = SHOW_CODE;
        lcd.clear();
```

```
        lcd.setCursor(0, 0);
        lcd.print("Enter code: 0991");
        lcd.setCursor(0, 1);
        lcd.print("into WushWash bot");
        lcd.setCursor(0, 3);
        lcd.print(prices[tempIndex]);
        lcd.print(" WUSH & ");
        lcd.print(temps[tempIndex]);

        String sendData = temps[tempIndex] + "," + String(prices[tempIndex]);
        mySerial.println(sendData);
        Serial.println(sendData);

        currentState = WAITING;
        updatePixyDisplay();
      }
      break;

    case WAITING:
      // handled earlier
      break;

    case STARTED:
      handleContinuousServo();
      break;



  }

  updatePixyDisplay();  // runs regardless of state


}

// 🌀 Continuous spinning back-and-forth
void handleContinuousServo() {
  updatePixyDisplay();
  static unsigned long lastSwitch = 0;
  static bool spinDirection = false;
  const unsigned long interval = 2000;

  if (millis() - lastSwitch >= interval) {
    if (spinDirection) {
      myServo.write(60);  // CW
    } else {
      myServo.write(120); // CCW
    }
    spinDirection = !spinDirection;
    lastSwitch = millis();
  }
}

void showWelcome() {
  updatePixyDisplay();
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("WELCOME TO WUSHWASH");
  lcd.setCursor(0, 1);
  lcd.print("Press START to begin");
}

void showSelection() {
```

```
  updatePixyDisplay();
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Temp: ");
  lcd.print(temps[tempIndex]);
  lcd.setCursor(0, 1);
  lcd.print("Price: RM");
  lcd.print(prices[tempIndex]);
  lcd.setCursor(0, 3);
  lcd.print("Press Next to cont");
}

void resetAll() {
  updatePixyDisplay();
  currentState = WELCOME;
  tempIndex = 0;
  myServo.write(88); // STOP servo
  showWelcome();
}

void updatePixyDisplay() {
  if (millis() - lastPixyUpdate < pixyInterval) return;
  lastPixyUpdate = millis();

  /* reset per-frame snapshot */
  currRed = currGreen = currBlue = currUnknown = 0;

  int numBlocks = pixy.getBlocks();

  if (numBlocks) {
    for (int i = 0; i < numBlocks; i++) {
      int sig = pixy.blocks[i].signature;
      if      (sig == 1) currRed++;
      else if (sig == 2) currGreen++;
      else if (sig == 3) currBlue++;
      else               currUnknown++;
    }
  }

  /* add only the *new* arrivals to the running total */
  if (currRed     > prevRed)     totalRed     += (currRed     - prevRed);
  if (currGreen   > prevGreen)   totalGreen   += (currGreen   - prevGreen);
  if (currBlue    > prevBlue)    totalBlue    += (currBlue    - prevBlue);
  if (currUnknown > prevUnknown) totalUnknown += (currUnknown - prevUnknown);

  /* remember this frame for the next comparison */
  prevRed     = currRed;
  prevGreen   = currGreen;
  prevBlue    = currBlue;
  prevUnknown = currUnknown;

  /* show the totals on row 2 (LCD index 2) */
  lcd.setCursor(0, 2);
  lcd.print("R:");
  lcd.print(totalRed);
  lcd.print(" G:");
  lcd.print(totalGreen);
  lcd.print(" B:");
  lcd.print(totalBlue);
  lcd.print("  ");          // clear stray chars if total shrinks (unlikely)
}
```

ESP32 Code

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <ArduinoJson.h>

const char* ssid = "pad6";
const char* password = "nazzhannn";

const char* firebaseApiKey = "AIzaSyCpIouGXycCudDVQXFCtSJxA82sfLG4chY";
const char* firebaseEmail = "muhammadnazhan34@gmail.com";
const char* firebasePassword = "nazhannazhan";

String idToken = "";
const char* projectId = "washingmachine-6f7d6";
const char* sessionId = "0991";

String tempReceived = "";
int priceReceived = 0;

void setup() {
  Serial.begin(115200);      // Serial Monitor
  Serial2.begin(9600, SERIAL_8N1, 16, 17);     // Serial to Arduino

  Serial.println("=== ESP32 SETUP START ===");

  WiFi.begin(ssid, password);
  Serial.print("Connecting to WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
  Serial.println("\nWiFi connected");

  if (signIn()) {
    Serial.println("Firebase signed in");
  } else {
    Serial.println("Firebase login failed");
  }

  Serial.println("=== SETUP COMPLETE ===");
}

void loop() {
  if (Serial2.available()) {
    String data = Serial2.readStringUntil('\n');
    data.trim();
    Serial.println("\nReceived from Arduino: " + data);

    int commaIndex = data.indexOf(',');
    if (commaIndex > 0) {
      tempReceived = data.substring(0, commaIndex);
      priceReceived = data.substring(commaIndex + 1).toInt();
      Serial.println("Temp: " + tempReceived + " | Price: RM" +
String(priceReceived));

      if (idToken != "") {
        updateFirestoreData(sessionId, tempReceived, priceReceived);
      } else {
```

```
        Serial.println("No Firebase token, skipping update");
      }
    } else {
      Serial.println("Invalid data format from Arduino");
    }
  }

  if (idToken != "") {
    String status = checkFirestoreStatus(sessionId);
    if (status == "paid") {
      Serial.println("paid");
      Serial2.println("paid");
    } else {
      Serial.println("Status: " + status);
      Serial2.println(status);
    }
  }

  delay(2000);
}

bool signIn() {
  Serial.println("Signing in to Firebase...");
  HTTPClient http;

http.begin("https://identitytoolkit.googleapis.com/v1/accounts:signInWithPassword?key=
" + String(firebaseApiKey));
  http.addHeader("Content-Type", "application/json");

  String payload = "{\"email\":\"" + String(firebaseEmail) + "\",\"password\":\"" +
String(firebasePassword) + "\",\"returnSecureToken\":true}";
  Serial.println("Auth Payload: " + payload);

  int httpCode = http.POST(payload);

  if (httpCode > 0) {
    String response = http.getString();
    Serial.println("Auth Response: " + response);

    DynamicJsonDocument doc(2048);
    deserializeJson(doc, response);
    idToken = doc["idToken"].as<String>();
    return true;
  } else {
    Serial.printf("Auth failed, HTTP code: %d\n", httpCode);
    Serial.println("Response: " + http.getString());
    return false;
  }
}

void updateFirestoreData(String sessionId, String temp, int price) {
  HTTPClient http;

  String url = "https://firestore.googleapis.com/v1/projects/" + String(projectId) +
               "/databases/(default)/documents/sessions/" + sessionId +
               "?updateMask.fieldPaths=temp&updateMask.fieldPaths=price";

  http.begin(url);
  http.addHeader("Content-Type", "application/json");
  http.addHeader("Authorization", "Bearer " + idToken);

  DynamicJsonDocument doc(512);
  JsonObject fields = doc.createNestedObject("fields");
```

```
  fields["temp"]["stringValue"] = temp;
  fields["price"]["integerValue"] = price;

  String jsonPayload;
  serializeJson(doc, jsonPayload);

  Serial.println("PATCH Firestore URL: " + url);
  Serial.println("Payload: " + jsonPayload);

  int httpCode = http.PATCH(jsonPayload);
  String response = http.getString();

  if (httpCode > 0) {
    Serial.println("Firestore update successful");
    Serial.println("Response: " + response);
  } else {
    Serial.printf("Firestore update failed: %d\n", httpCode);
    Serial.println("Response: " + response);
  }

  http.end();
}

String checkFirestoreStatus(String sessionId) {
  HTTPClient http;
  String url = "https://firestore.googleapis.com/v1/projects/" + String(projectId) +
               "/databases/(default)/documents/sessions/" + sessionId;

  http.begin(url);
  http.addHeader("Authorization", "Bearer " + idToken);

  int httpCode = http.GET();
  String payload = http.getString();

  if (httpCode > 0) {
    Serial.println("GET status response: " + payload);

    DynamicJsonDocument doc(2048);
    deserializeJson(doc, payload);
    String status = doc["fields"]["status"]["stringValue"];
    return status;
  } else {
    Serial.printf("GET failed: %d\n", httpCode);
    Serial.println("Response: " + payload);
    return "";
  }

  http.end();
}
```

Google App Script

Code.gs

```
// Configuration
var apiToken = "8082420458:AAG1RmonYdTZH9gHFXW48SNWvqwO9APSwuM";
var appUrl   =
"https://script.google.com/macros/s/AKfycbxtXuEaGCtGpRb-IaSItvHQowcV653DgIyE_BbGde1d37
eZPYBtMKWKB5wpqEIPst9t/exec";
var apiUrl   = "https://api.telegram.org/bot" + apiToken;

// Set webhook
function setWebhook() {
  var url = apiUrl + "/setwebhook?url=" + appUrl;
  var res = UrlFetchApp.fetch(url).getContentText();
  Logger.log(res);
}

// Handle webhook
function doPost(e) {
  var webhookData = JSON.parse(e.postData.contents);
  var chatId = webhookData.message.from.id.toString();
  var text = webhookData.message.text.trim();
  var props = PropertiesService.getScriptProperties();
  var state = props.getProperty(chatId);

  if (text === "/start") {
    props.setProperty(chatId, "awaiting_matric");
    sendMessage(chatId, "👋 Welcome to <b>WushWash</b>!\nPlease enter your matric
number:");
  }

  else if (state === "awaiting_matric") {
    var matric = text;
    var user = readFirestoreUser(matric);

    if (user) {
      var name = user.fields.name.stringValue;
      var balance = parseInt(user.fields.balance.integerValue);
      var password = user.fields.password.stringValue;

      props.setProperty(chatId, "login_password");
      props.setProperty("matric_" + chatId, matric);

      sendMessage(chatId, `👤 Hello <b>${name}</b>!\n\nPlease enter your password 🔐
to sign in:`);
    } else {
      props.setProperty(chatId, "registering_matric");
      sendMessage(chatId, "❌ Matric number not found.\nPlease enter your matric
number to register:");
    }
  }

  else if (state === "registering_matric") {
```

```
    props.setProperty("matric_" + chatId, text);
    props.setProperty(chatId, "registering_name");
    sendMessage(chatId, "Please enter your name:");
  }

  else if (state === "registering_name") {
    props.setProperty("user_name_" + chatId, text);
    props.setProperty(chatId, "registering_password");
    sendMessage(chatId, "Please enter your password (minimum 6 characters):");
  }

  else if (state === "registering_password") {
    var password = text;
    var matric = props.getProperty("matric_" + chatId);
    var name = props.getProperty("user_name_" + chatId);

    if (password.length < 6) {
      sendMessage(chatId, "❌ Password must be at least 6 characters. Please try
again.");
    } else {
      createUserWithPassword(matric, name, password);
      props.deleteProperty(chatId);
      sendMessage(chatId, "✅ Account created successfully! You get 10 Wush! Please
press /start to begin using the machine.");
    }
  }

  else if (state === "login_password") {
    var password = text;
    var matric = props.getProperty("matric_" + chatId);
    var user = readFirestoreUser(matric);

    if (user && user.fields.password.stringValue === password) {
      var balance = parseInt(user.fields.balance.integerValue);
      props.setProperty(chatId, "logged_in");
      sendMessage(chatId, `✅ Password correct!\n💰 Your balance: <b>${balance}
Wush</b>`);
      sendMessage(chatId, `Now you can:\n🔢 Press /entercode to enter your machine
code\n\n💸 Press /topup to top up your balance`);
    } else {
      sendMessage(chatId, "❌ Incorrect password. Please try again:");
    }
  }

  else if (text === "/topup" && state === "logged_in") {
    var matric = props.getProperty("matric_" + chatId);
    var user = readFirestoreUser(matric);
    if (user) {
      var balance = parseInt(user.fields.balance.integerValue);
      props.setProperty(chatId, "topup_amount");
      sendMessage(chatId, `💰 Your current balance is: <b>${balance} Wush</b>\n\nHow
much would you like to top up? Please enter the amount.`);
    } else {
      sendMessage(chatId, "❌ User not found.");
```

```
    }
  }

  else if (text === "/entercode" && state === "logged_in") {
    props.setProperty(chatId, "awaiting_code");
    sendMessage(chatId, "🔢 Please enter your machine code:");
  }

  else if (state === "awaiting_code") {
    var code = text;
    var session = readFirestoreSession(code);

    if (session) {
      var price = parseInt(session.fields.price.integerValue);
      var temp = session.fields.temp.stringValue;
      props.setProperty("session_code_" + chatId, code);
      props.setProperty("price_" + chatId, price);
      props.setProperty(chatId, "ready_to_pay");
      sendMessage(chatId, `✅ Code valid!\n💸 Machine price: <b>${price}
Wush</b>\n🌡️Temperature selected: <b>${temp}</b> \n\nPress /pay to continue with
payment.`);
    } else {
      sendMessage(chatId, "❌ Invalid code. Please try again:");
    }
  }

  else if (text === "/pay" && state === "ready_to_pay") {
    var sessionCode = props.getProperty("session_code_" + chatId);
    var price = parseInt(props.getProperty("price_" + chatId));
    var matric = props.getProperty("matric_" + chatId);
    var user = readFirestoreUser(matric);

    if (user) {
      var currentBalance = parseInt(user.fields.balance.integerValue);

      if (currentBalance >= price) {
        var newBalance = currentBalance - price;
        updateUserBalance(matric, newBalance);
        updateSessionStatus(sessionCode, "paid");

        props.setProperty(chatId, "session_paid");

        sendMessage(chatId, `✅ Payment successful!\n🍩 Starting the machine...\n💰
Remaining balance: <b>${newBalance} Wush</b>\n\n✅ When you have collected your item,
press /done 👋`);
      } else {
        sendMessage(chatId, "⚠️ Insufficient balance. Please top up your Wush.
/topup");
      }
    } else {
      sendMessage(chatId, "❌ User not found.");
    }
  }
```

```
  else if (text === "/done" && state === "session_paid") {
    var sessionCode = props.getProperty("session_code_" + chatId);
    updateSessionStatus(sessionCode, "unpaid");
    props.deleteProperty(chatId);
    props.deleteProperty("session_code_" + chatId);
    props.deleteProperty("price_" + chatId);
    props.deleteProperty("matric_" + chatId);
    sendMessage(chatId, "👋 Bye! Thank you for using <b>WushWash</b>!\nHave a clean
day! 👦🏷️\nPress /start to WushWash!👦");
  }

  else if (state === "topup_amount") {
    var amount = parseInt(text);

    if (isNaN(amount) || amount <= 0) {
      sendMessage(chatId, "❌ Invalid top-up amount. Please enter a valid number.");
    } else {
      props.setProperty(chatId, "topup_tac");
      props.setProperty("topup_amount_" + chatId, amount);
      sendMessage(chatId, `✅ You are about to top up <b>${amount} Wush</b>.\nPlease
enter the TAC code sent to your phone number (+601****5423).\n\nFor this prototype,
use the code: <b>246357</b>`);
    }
  }

  else if (state === "topup_tac") {
    var tac = text.trim();
    var amount = parseInt(props.getProperty("topup_amount_" + chatId));

    if (tac !== "246357") {
      sendMessage(chatId, "❌ Invalid TAC code. Please enter the correct code.");
    } else {
      var matric = props.getProperty("matric_" + chatId);
      var user = readFirestoreUser(matric);
      if (user) {
        var newBalance = parseInt(user.fields.balance.integerValue) + amount;
        updateUserBalance(matric, newBalance);
        props.deleteProperty(chatId);
        props.deleteProperty("topup_amount_" + chatId);
        props.deleteProperty("topup_tac_" + chatId);
        sendMessage(chatId, `✅ Top-up successful! Your new balance is
<b>${newBalance} Wush</b>.\n\nPress /start to begin again.`);
      } else {
        sendMessage(chatId, "❌ User not found.");
      }
    }
  }

  else if (text === "/help") {
    doHelpCommand(chatId);
  }

  else {
    sendMessage(chatId, "❓ Command not found.\nUse /start to begin.");
```

```
  }
}

function doGet(e) {
  return ContentService.createTextOutput("Method GET not allowed");
}

function sendMessage(chatId, text) {
  var url = apiUrl + "/sendMessage?parse_mode=HTML&chat_id=" + chatId + "&text=" +
encodeURIComponent(text);
  UrlFetchApp.fetch(url, { muteHttpExceptions: true });
}

function doHelpCommand(chatId) {
  const helpText = `
  *Welcome to the WushWash Bot!*

  Here are the available commands you can use:

  /start - Start the washing machine process.
  /help - Show this help message.

  Please call +60123456789 for enquiries.

  If you need further assistance, feel free to ask!`;

  sendMessage(chatId, helpText);
}
```

firebase.js.gs

```
// Paste your service account JSON here
var serviceAccount = {
  "type": "service_account",
  "project_id": "washingmachine-6f7d6",
  "private_key_id": "4e09807e51ba3e6727cc2b08e565c3d39af69ec1",
  "private_key": "-----BEGIN PRIVATE
KEY-----\nMIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQCSOyR2UUNQpcR3\nGdmpYhzPm8
a/wdKleypVqfrG7q7ICsD3Q46hTLrO0Dc+s5kj3n6AA5+8AHMIwGEa\nP6xQtDuelLQpdZq/O2F41bQpfWU22J
oKvejPgxUKd54AAUOucU+7SwhcB1hfTY07\npofebFbVv4KCGG6v6jOdeK82Go3k/EKApVH4NmrN94TNgur397
PNIAtVGG2X/iv3\ntRvZ5a/R3A3gE9V/WyIOgQyevcQqYpb53EYjwg41ARCameNDp4bDub6AwZeQH294\nyrcs
GjYyurgVYbXfwshvR8QqTP64KgVMgmLBp/wL8hflz4BDvUzPxDThFvL9i7zo\nW0IlG79rAgMBAAECggEAR46V
ayic7iwQFdAaiR6vxpAOE+v+7md08fJ5+zr1YIKS\npuzyMNBhYa7WOeKc/8Zy1KXxVohJwnCRk8ywv9VSj2lp
OBiKnyWprv20adEo5Eve\nJX+FG3ajpzmdoZb8zbI5OAehkZcukYRbpmL5fcg6PeK3NH5MO905YMebxbf80544
\nUGfehAqlWL+usGLQAlaAazGEUEFyYw2faif50q/2wfBvOcVhoYW9G9Bb3zqEIZwY\nj9QhCqtfxYVQmihOLo
wNucNd+UfaG/MNr/MkO/mzuLaG6j2sSJixcxJ+oQrEv5Nw\nWSyYkkxAVApvVTQMWt92TT7Hc0dDOlOdjbH78y
cNMQKBgQDJhMwDAURGXidQvYHn\nxlsjmbCxs9XaBP9sGT/VDlBGW6jCM+4nnFSu+8Se2bNNDTYyKHLjDxAJ11
1ULfYj\nahuwh2Mn6uU8zL5VJhm7490v69S/YNUrwW5ZN8g9vUk14wRyBj1Ld4r9KnionKo+\n7G+sMBKMcPgo
pXKCuNnRzfdglwKBgQC5w9udtIGDe1De0w2jQB6oGmLipN694K5c\nkWZJFsPPl1GCGINmJgUKH1IerpdEM0iB
x33pQsyYazOEO0S05BTQpj3oh2nr95d/\nrWHfhqV4cJzzQUra4jaowb186hLz962Ap4Fw5PO0dxfMHAyTpkzF
m/cqEu0quoQ5\n51jf2O4eTQKBgC0SWJbrd5vNfzmdySpPwR6rR0vXjyMeLwq8cvxyIR47bflRqBiP\n7Jjekt
F8m4/GPrLlPRzvx4SL+MHVLdIp71uCTHknHnKE0KU2ylat1lDbjE/ploQc\n1VN7GHxGk9NLnTjrkFYKvLl3AF
mmAKXmN+vpLJzgv4BH0N7P/Vfhm0ytAoGAMWts\nkqEBz5PP6Oem+iZFwZz7b6ZC56rNWC0KVCltQF3WPlD9LA
CzWIUKmMUD0w39nSEQ\nWyz9NxNV6jjGR4ViEDwZvjHAORluQXZD0m6EuOzkpEZbfq2tsumP2pp/Hk5m5xIo\n
+nq+ZuVD/HX91o3oYK3kbFoKpFJoeOsWFr8kYgUCgYEAn3oF5r9jSrT6+4ssr2sT\n/zGgNCJQFcsPlGXMnvwS
g1EqaAzuUXm8lvRYox8SqBCk0GxItBSRGLKfotU3iGXd\nxIzN29geYo8d1NG5K4DABerdUNpFEig7cpAnVMR4
on2g3UqvEkQBVmeYN7jyoLy4\n668yqZ5K72ze9RntO2RnAto=\n-----END PRIVATE KEY-----\n",
  "client_email": "telegram-bot-access@washingmachine-6f7d6.iam.gserviceaccount.com",
  "client_id": "102710913890327596351",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleapis.com/token",
  "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
  "client_x509_cert_url":
"https://www.googleapis.com/robot/v1/metadata/x509/telegram-bot-access%40washingmachin
e-6f7d6.iam.gserviceaccount.com",
  "universe_domain": "googleapis.com"
};

// Generate Access Token using JWT
function getAccessToken() {
  const url = "https://oauth2.googleapis.com/token";
  const payload = {
    grant_type: "urn:ietf:params:oauth:grant-type:jwt-bearer",
    assertion: createJwt()
  };
  const options = {
    method: "post",
    payload: payload,
    muteHttpExceptions: true
  };
  const response = UrlFetchApp.fetch(url, options);
  const result = JSON.parse(response.getContentText());
```

```
  return result.access_token;
}

// Create JWT from service account
function createJwt() {
  const header = {
    alg: "RS256",
    typ: "JWT"
  };
  const now = Math.floor(Date.now() / 1000);
  const payload = {
    iss: serviceAccount.client_email,
    scope: "https://www.googleapis.com/auth/datastore",
    aud: "https://oauth2.googleapis.com/token",
    iat: now,
    exp: now + 3600
  };
  const signatureInput = Utilities.base64EncodeWebSafe(JSON.stringify(header)) + "." +
                         Utilities.base64EncodeWebSafe(JSON.stringify(payload));
  const signature = Utilities.base64EncodeWebSafe(
    Utilities.computeRsaSha256Signature(signatureInput, serviceAccount.private_key)
  );
  return signatureInput + "." + signature;
}

// 🔍 Get user document by matric
function readFirestoreUser(matric) {
  const accessToken = getAccessToken();
  const url =
`https://firestore.googleapis.com/v1/projects/${serviceAccount.project_id}/databases/(
default)/documents/users/${matric}`;
  const options = {
    headers: {
      Authorization: `Bearer ${accessToken}`
    },
    muteHttpExceptions: true
  };
  const response = UrlFetchApp.fetch(url, options);
  return response.getResponseCode() === 200 ? JSON.parse(response.getContentText()) :
null;
}

// 🧾 Create user with password and default balance
function createUserWithPassword(matric, name, password) {
  const accessToken = getAccessToken();
  const url =
`https://firestore.googleapis.com/v1/projects/${serviceAccount.project_id}/databases/(
default)/documents/users?documentId=${matric}`;
  const payload = JSON.stringify({
    fields: {
      name: { stringValue: name },
      balance: { integerValue: "10" }, // must be string for REST API
      password: { stringValue: password }
    }
```

```
  });
  const options = {
    method: "POST",
    contentType: "application/json",
    payload: payload,
    headers: {
      Authorization: `Bearer ${accessToken}`
    }
  };
  const response = UrlFetchApp.fetch(url, options);
  return response.getContentText();
}

//update user balance
function updateUserBalance(matric, newBalance) {
  const accessToken = getAccessToken();
  const url =
`https://firestore.googleapis.com/v1/projects/${serviceAccount.project_id}/databases/(
default)/documents/users/${matric}?updateMask.fieldPaths=balance`;
  const payload = JSON.stringify({
    fields: {
      balance: { integerValue: String(newBalance) }
    }
  });
  const options = {
    method: "PATCH",
    contentType: "application/json",
    payload: payload,
    headers: {
      Authorization: `Bearer ${accessToken}`
    }
  };
  const response = UrlFetchApp.fetch(url, options);
  return response.getContentText();
}

// 🔴 Top-up balance function
function topUpBalance(matric, amount) {
  const user = readFirestoreUser(matric);
  if (user) {
    const currentBalance = parseInt(user.fields.balance.integerValue, 10);
    const newBalance = currentBalance + amount; // Add the top-up amount to current
balance
    return updateUserBalance(matric, newBalance); // Update Firestore with new balance
  } else {
    return "User not found";
  }
}

// 🔍 Get session document by code
function readFirestoreSession(code) {
  const accessToken = getAccessToken();
```

```
  const url =
`https://firestore.googleapis.com/v1/projects/${serviceAccount.project_id}/databases/(
default)/documents/sessions/${code}`;
  const options = {
    headers: {
      Authorization: `Bearer ${accessToken}`
    },
    muteHttpExceptions: true
  };
  const response = UrlFetchApp.fetch(url, options);
  return response.getResponseCode() === 200 ? JSON.parse(response.getContentText()) :
null;
}

// 🧾 Update session status field
function updateSessionStatus(code, newStatus) {
  const accessToken = getAccessToken();
  const url =
`https://firestore.googleapis.com/v1/projects/${serviceAccount.project_id}/databases/(
default)/documents/sessions/${code}?updateMask.fieldPaths=status`;
  const payload = JSON.stringify({
    fields: {
      status: { stringValue: newStatus }
    }
  });
  const options = {
    method: "PATCH",
    contentType: "application/json",
    payload: payload,
    headers: {
      Authorization: `Bearer ${accessToken}`
    }
  };
  const response = UrlFetchApp.fetch(url, options);
  return response.getContentText();
}
```

Student's Declaration

Certificate of Originality and Authenticity

This is to certify that we are responsible for the work submitted in this report, that the original work is our own except as specified in the references and acknowledgement, and that the original work contained herein have not been untaken or done by unspecified sources or persons.

We hereby certify that this report has not been done by only one individual and all of us have contributed to the report. The length of contribution to the reports by each individual is noted within this certificate.

We also hereby certify that we have read and understand the content of the total report and no further improvement on the reports is needed from any of the individual's contributors to the report.

We therefore, agreed unanimously that this report shall be submitted for marking and this final printed report has been verified by us.